Resolution Decides the Guarded Fragment

Hans de Nivelle

April 6, 1998

Contents

Abstract

We give resolution based decision procedures for the guarded fragment of ([ANB96]), and for the loosely guarded fragment of ([vBenthem97]). The relevance of the guarded fragments lies in the fact that many modal logics can be translated into them. In this way the guarded fragments act as a framework explaining the nice properties of these modal logics. By constructing an implementable decision procedure for the guarded fragments we define an effective procedure for deciding these modal logics.

1 Introduction

The guarded fragment was inspired by the following observation:

- Many modal propositional modal logics have very good properties, they are decidable, have the finite model property, and interpolation.

- These modal logics can be translated into first order logic, using a standard (relational) translation based on the Kripke frames.

$$\begin{array}{lll} \Box A & \Rightarrow & \forall s' R(s,s') \rightarrow \cdots \\ \Diamond A & \Rightarrow & \exists s' R(s,s') \wedge \cdots \end{array}$$

The fragment of first order formulae that can be a translation of a modal formula must also have these properties. This leads to the following question: What makes this set of translations of modal formulae so nice? One explanation could be the fact that modal logic translates into the 2-variable fragment, which is decidable. This is not sufficient. The logic K can be translated into the 2-variable fragment, but most other modal logics cannot. Also the 2-variable fragment lacks interpolation, although it is decidable.
The guarded fragment is based on the observation that in the translations all quantifiers are *conditional* in an accessibility condition, i.e. they all have form: for all worlds s' for which $R(s,s')$, something holds in s'. This leads to a definition of a the *guarded fragment* in which all universal quantifiers occur as $\forall \overline{x}(G(\overline{x},\overline{y}) \rightarrow \Phi(\overline{x},\overline{y})\ )$. It turns out that this fragment, although it still cannot explain all modal logics, has good model theoretic properties. There are more perspectives for generalization than from the 2-variable fragment, as the 3-variable fragment is already undecidable.
An example of such a generalization is the loosely guarded fragment, which we will also show decidable by resolution.
In this report we develop resolution decision procedures for the guarded fragment and the loosely guarded fragment. We define guarded clauses, and show that first order guarded formulae can be translated into sets of guarded clauses. After that we show that sets of guarded clause sets are decidable using techniques that are standard in the field of resolution decision procedures. The restriction of resolution that has to be used is based on a so called *ordering refinement*. All of the major theorem provers (SPASS, [Weid96], OTTER [McCune95], and Gandalf, [ATP97]) support orderings, although not exactly the one that we need. Therefore it is reasonable to expect that the ordering refinement that we need can be built in to existing provers without much effort. In order to decide the loosely guarded fragment we define a similar notion of loosely guarded clause. However deciding sets of loosely guarded clauses is much less straightforward, than deciding sets of guarded clauses: We need a non-trivial modification of hyperresolution on top of the ordering refinement to decide the loosely guarded fragment. In order to prove its completeness an extension of the resolution game has been necessary.
With an effective resolution based decision procedure, implementation has become feasable. The necessary techniques are standard in the field of automated theorem proving. An implementation would be useful since it would act as a framework for the decision of modal logics.

In this way it even improves the resolution methods of ([Niv93]), since the resolution methods given there are impractical, and they do not even provide a decision procedure for many decidable modal logics.

1.1 First Order Logic

In this section we briefly introduce first order logic/interpretations. The main purpose is to introduce our notation.

Definition 1.1 We define the language of *first order logic*: We assume a fixed set F of function/constant symbols, and a fixed set P of predicate symbols. We define the set of terms *terms* as the set of objects that can be built by finitely often applying the following rule:

- If $t_1, \dots, t_n$ are terms, with $n \geq 0$, and $f \in F$, then $f(t_1, \dots, t_n)$ is a term.

Terms of the form $f(\ )$ are called *constants* and the brackets are omitted. If $n > 0$, then the term is called *functional*.

The set of *formulae* is the set of objects that can be built by finitely often applying the following rules:

- If $t_1, \dots, t_n$ are terms, and $n \geq 0$, then $p(t_1, \dots, t_n)$ is a formula. If $n = 0$, then $p(\ )$ is called *propositional*, and the brackets are omitted. Formulae of the form $p(t_1, \dots, t_n)$ are called *atoms*.
- If A is a formula, then $\neg A$ is a formula. If A is an atom, then $\neg A$ is called a *literal*.
- $\perp$ and $\top$ are formulae.
- If A and B are formulae, then so are $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$.
- If A is a formula, $v \in F$, then $\forall v A$ and $\exists v A$ are formulae.

We identify variables and zero arity function symbols. We allow the same function/predicate symbol to occur with different arities. We have to pay a small price in the definition of interpretations for this. We cannot just interpret the function symbols, but we have to interpret them separately for each arity.

Definition 1.2 Let F be a first order formula. An interpretation I of F is an ordered structure (D, π_i, ϕ_j). Here D is a domain. The π_i and the ϕ_j are partial functions. The π_i interpret the predicate symbols of arity i. The ϕ_j interpret the function symbols of arity j.

- D is a nonempty set, called the *domain* of I.

-
  - For every $f \in F$, for which there occurs a term $f(t_1, \ldots, t_n)$ with $n > 0$ in F, $\phi_n(f)$ is defined and equals a function from D^n to D.
  - For every $f \in F$ that is free in F, $\phi_0(f)$ is defined and equals an element of D.
  - For every $p \in P$, for which there is an atom $p(t_1, \ldots, t_n)$ in F, $\pi_n(p)$ is defined and equals a function from D^n to $\{f, t\}$.

Definition 1.3 We extend the interpretation I to a function $[\ ]$ from terms and from first order formulae. For a term t, the result $[t]$ equals an element of D. For a formula F, the result $[F]$ is an element of $\{f, t\}$.

For the quantifiers we need the following: Let ϕ be a partial function from some domain D_1 to some range D_2. If $d_1 \in D_1$, and $d_2 \in D_2$ then $\phi(d_1 := d_2)$ is the function ϕ' for which $d \neq d_1 \Rightarrow \phi'(d) = \phi(d)$, and for which $\phi'(d_1) = d_2$.

If F is a first order formula, and $I = (D, \pi_i, \phi_j)$ is an interpretation of F, then:

- For each term $f(t_1, \ldots, t_n)$, the value of $[f(t_1, \ldots, t_n)]$ is defined as: $\phi_n(f)([t_1], \ldots, [t_n])$.
- For each atom $p(t_1, \ldots, t_n)$, the value of $[p(t_1, \ldots, t_n)]$ is defined as $\pi_n(p)([t_1], \ldots, [t_n])$.
- $[\top] = t$, and $[\bot] = f$.
- For each formula of the form $A \vee B$, $[A \vee B] = t$ if either $[A] = t$, or $[B] = t$. Otherwise $[A \vee B] = f$.
- The extension rules for the operators $\neg, \wedge, \leftarrow, \leftrightarrow$ are as usual.
- For each formula of the form $\forall v A$, the result $[\forall v A]$ equals t if for all $d \in D$, for the extension $< >$ of the interpretation $I' = (D, \pi_i, \phi_0[v := d], \phi_1, \ldots, \phi_j, \ldots)$, it is the case that $< A >= t$.
- For each formula of the form $\exists v A$, the result $[\exists v A]$ equals t if there exists a $d \in D$, such that for the extension $< >$ of the interpretation $I' = (D, \pi_i, \phi_0[v := d], \phi_1, \ldots, \phi_j, \ldots)$, it is the case that $< A >= t$.

1.2 The Resolution Calculus

We briefly introduce the resolution calculus. Contrary to the case of first order logic, we distinguish constants and variables. This makes it possible to omit the $\forall$-quantifiers in clauses.

Definition 1.4 We assume a fixed set of functions/constant symbols F, a fixed set of predicate/propositional symbols P, and a fixed set of variables V. The set of *terms* is recursively defined as follows:

1. A variable is a term.
2. If $t_1, \ldots, t_n$, with $n \geq 0$, are terms, and $f \in F$, then $f(t_1, \ldots, t_n)$ is a term.

If $t_1, \ldots, t_n$, with $n \geq 0$, are terms, and $p \in P$, then $p(t_1, \ldots, t_n)$ is an *atom*.

A *literal* is an atom A, or its negation $\neg A$. Atoms of the form A are called *positive*. Atoms of the form $\neg A$ are called *negative*.

A *clause* is a finite set of literals.

A term that contains no variables is called *ground*. A term of the form c is called *constant*. A term of the form $f(t_1, \ldots, t_n)$, with $n > 0$, is called *functional*.

Definition 1.5 We define some complexity measures for atoms/clauses/literals:

- Let A be an atom. The *depth* of A is recursively defined as follows:
  - If A is a variable, then $\text{Depth}(A) = 1$.
  - $\text{Depth}(f(t_1, \ldots, t_n))$ equals the maximum of $\{1, 1+\text{Depth}(t_1), \ldots, 1+\text{Depth}(t_n)\}$.

  The depth of a literal equals the maximal depth of its atoms. The depth of a clause c equals the maximal depth of a literal in c, or 0 for the empty clause.
- Let A be an atom. The *vardepth* of A is recursively defined as follows:
  - If A is ground, then $\text{Vardepth}(A) = -1$.
  - If A is a variable, then $\text{Vardepth}(A) = 0$.
  - Otherwise $\text{Vardepth}(f(t_1, \ldots, t_n)$ equals the maximum of $\{1 + \text{Vardepth}(t_1), \ldots, 1 + \text{Vardepth}(t_n)\}$.

  The vardepth of a literal equals the vardepth of its atom. The vardepth of a clause c equals the maximal depth of a literal in c. The vardepth of the empty clause is defined as -1. The vardepth of a set of clauses C is defined as the maximal vardepth of a clause in C.
- Let A be an atom/literal/clause. $\text{Var}(a)$ is defined as the set of variables that occur in A.

- Let c be a clause. Varnr(c) equals the size of Var(c). If C is a set of clauses, then Varnr(C) equals maximal number of variables that occurs in a clause of C.
- Let A be a literal. The *complexity* of A, written as $\#A$ equals the total number of function/constant/variable occurrences in it.

Definition 1.6 A *substitution* is a finite set of variable assignments of the form $\{V_1 := t_1, \ldots, V_n := t_n\}$, such that $V_i \neq t_i$, and $V_i = V_j \Rightarrow t_i = t_j$. The first conditions ensures non-redundancy, the second condition ensures consistency.

We write $A\Theta$ for the effect of Θ on term A.

If Θ_1 and Θ_2 are substitutions, then the *composition* of Θ_1 and Θ_2 is defined as the substitution $\{v := v\Theta_1\Theta_2| \; |v \neq v\Theta_1\Theta_2\}$.

For two literals A and B a unifier is a substitution Θ, such that $A\Theta = B\Theta$. A *most general unifier* Θ is a substitution such that $A\Theta = B\Theta$, and

$$\forall \Theta'(A\Theta' = B\Theta') \Rightarrow \exists \Sigma(\Theta' = \Theta \cdot \Sigma).$$

For the composition holds, for all terms $A(\Theta_1 \cdot \Theta_2) = (A\Theta_1)\Theta_2$.
The notion of mgu was introduced by J. A. Robinson in ([Robinson65], and is of extreme importance for the field of theorem proving. The impact of the mgu on theorem proving can be characterized as follows: The substitutions used in an unsatisfiable set of Herbrand instances are completely determined by the conflicting pairs in the Herbrand instance. This means that the search procedure can concentrate on searching conflicting pairs instead of enumerating the Herbrand base.
We present an algorithm for computing the mgu. The algorithm is not efficient because its purpose is proving properties of mgu's, rather than applying it.

Definition 1.7 The following algorithm effectively decides whether or not two literals A and B have a unifier:

Let A and B be two literals, such that $A \neq B$. A *minimal difference* is obtained as follows:

1. Put $A' = A$, and $B' = B$.
2. As long as A' has form $p(t_1, \ldots, t_n)$ and B' has form $p(u_1, \ldots, u_n)$, replace A' by t_i and B' by u_i, for an i, such that $t_i \neq u_i$.

The algorithm is defined as follows: Let A and B be the terms to be unified. Put $\Theta = \{\ \}$, the idempotent substitution.

1. If $A = B$, then Θ equals the most general unifier.

2. As long as $A \neq B$, let (A', B') be a minimal difference. Then:
   - If (A', B') has form $(p(t_1, \ldots, t_n),\ q(u_1, \ldots, u_m),)$ with $p \neq q$, or $n \neq m$, then report failure.
   - If (A', B') has form (V, t), where V is a variable, $V \neq t$, but V occurs in t, then report failure.
   - If (A', B') has form (t, V), where V is a variable, $V \neq t$, but V occurs in t, then report failure.
   - If (A', B') has form (V, t) where V is a variable, and V does not occur in t, then put
     $A := A\{V := t\}$, $B := B\{V := t\}$, $\Theta := \Theta \cdot \{V := t\}$.
   - If (A', B') has form (t, V), where V is a variable, and V does not occur in t, then put
     $A := A\{V := t\}$, $B := B\{V := t\}$, $\Theta := \Theta \cdot \{V := t\}$.

Theorem 1.8 The procedure of Definition 1.7 is complete and sound, and strongly normalizing. The last property means that the outcome does not depend, modulo renaming substitutions, on the minimal difference that is selected.

Definition 1.9 We define the ordered resolution rule, and factorization rule: Let $\sqsubset$ be an order on literals.

Res Let $\{A_1\} \cup R_1$ and $\{\neg\ A_2\} \cup R_2$, be two clauses, s.t.

- $\{A_1\} \cup R_1$ and $\{A_2\} \cup R_2$ have no variables in common,
- For no $A \in R_1$, it is the case that $A_1 \sqsubset A$,
- For no $A \in R_2$, it is the case that $A_2 \sqsubset A$,
- A_1 and A_2 have mgu Θ.

Then the clause $R_1\Theta \cup R_2\Theta$ is called a *resolvent*.

Let $\{A_1, A_2\} \cup R$ be a clause, such that A_1 and A_2 have an mgu Θ. The clause $\{A_1\Theta\} \cup R\Theta$ is called a *factor* of $\{A_1, A_2\} \cup R$.

It is well-known that resolution together with factoring is a complete refutation procedure for sets of clauses. There is a proof in ([Robinson65]), or see ([Niv95]) for a more modern proof.

2 Building Blocks

In this section we discuss the building blocks from which the resolution decision procedures in the next section are built. First we discuss transformations from first order logic to clauses. Clausal logic is only a subset of first order logic. In order to deal with full first order logic we need translation techniques from first order logic into clausal logic. There are different manners to do this, and not with every manner we can obtain a decision procedure.

After that we dicuss special properties of literals. In order to obtain termination for resolution it is necessary that the literals in the clause sets have a special form: They must be *weakly covering*. Although the definition of this notion is technical, covering literals arise naturally from the Skolemization of decidable classes. We show that covering literals are preserved under resolution, and that the mgu of two covering literals is not larger than the largest of them.

2.1 Transformation to Clausal Normal Form

We give a collection of operators that can be used for the transformation from first order logic to clausal logic. The first step consists of the elimination of $\leftarrow$ and $\leftrightarrow$. We define all operators as working on sets of formulae rather than on formulae themselves, because operators may split one formula into different formulae.

Definition 2.1 Let $C = \{F_1, \ldots, F_n\}$ be a set of formulae. We define $\mathrm{Na}(C)$ as the result of making the following replacements in the F_i :

$$\begin{array}{lcl} A \leftrightarrow B & \Rightarrow & (\neg\, A \vee B) \wedge (\neg\, B \vee A), \\ A \rightarrow B & \Rightarrow & \neg\, A \vee B. \end{array}$$

The negation normal form is obtained by moving the negations inward as far as possible. This makes it possible to immediately see the sign of an occurrence of a subformula.

Definition 2.2 Let $C = \{F_1, \ldots, F_n\}$ be a set of formulae not containing $\leftarrow$ or $\leftrightarrow$. $\mathrm{NNF}(C)$ is the set of formulae that is obtained by normalizing the F_i under the following rewrite system:

$$\begin{array}{lcl} \neg\, (A \vee B) & \Rightarrow & \neg\, A \wedge \neg\, B, \\ \neg\, (A \wedge B) & \Rightarrow & \neg\, A \vee \neg\, B, \\ \neg\, \forall v A & \Rightarrow & \exists v\, \neg\, A, \\ \neg\, \exists v A & \Rightarrow & \forall v\, \neg\, A, \\ \neg\, \neg\, A & \Rightarrow & A, \\ \neg\, \bot & \Rightarrow & \top, \\ \neg\, \top & \Rightarrow & \bot. \end{array}$$

$$\begin{array}{lcllcl} (A \vee \bot) & \Rightarrow & A, & (A \vee \top) & \Rightarrow & \top, \\ (\bot \vee A) & \Rightarrow & A, & (\top \vee A) & \Rightarrow & \top, \\ (A \wedge \bot) & \Rightarrow & \bot, & (A \wedge \top) & \Rightarrow & A, \\ (\bot \wedge A) & \Rightarrow & \bot, & (\top \wedge A) & \Rightarrow & A, \\ (\forall v \bot) & \Rightarrow & \bot, & (\forall v \top) & \Rightarrow & \top, \\ (\exists v \bot) & \Rightarrow & \bot, & (\exists v \top) & \Rightarrow & \top. \end{array}$$

The purpose of the following operator is simplification. Red tries to simplify a formula by moving quantifiers inward as far as possible. This may lead to less complex Skolem terms.

Definition 2.3 Let $C = \{F_1, \ldots, F_n\}$ be a set of formulae. We define $\mathrm{Red}(C)$ as the result of normalizing the F_i using the following rewrite rules:

- If x is not free in A, then
$$Qx(A \wedge B) \Rightarrow A \wedge QxB,$$
and
$$Qx(A \vee B) \Rightarrow A \vee QxB.$$
- If x is not free in B, then
$$Qx(A \wedge B) \Rightarrow (QxA) \wedge B,$$
and
$$Qx(A \vee B) \Rightarrow (QxA) \vee B.$$

In the book ([CL73]), the following normal form is used in the process of the translation. It is not a good one, since it increases the complexity of the Skolem terms. Only for the sake of completeness we mention it.

Definition 2.4 Let $C = \{F_1, \ldots, F_n\}$ be a set of formulae. $\mathrm{Prenex}(C)$ is defined as the set that is the result of applying the following rewrite rules:

$$\begin{array}{lcl} (QxA) \vee B & \Rightarrow & Qx(A \vee B), \\ (QxA) \wedge B & \Rightarrow & Qx(A \wedge B), \\ B \vee QxA & \Rightarrow & Qx(B \vee A), \\ B \wedge QxA & \Rightarrow & Qx(B \wedge A). \end{array}$$

All rules have the condition that x is not free in B. If x is free in B, then x has to be replaced first by a new variable y that is not free in B and not free in A, as follows:
$$QxA \Rightarrow QyA[x := y].$$

We now come to structural transformations. Structural transformations replace certain subformulae by fresh names, together with a definition of the name. Structural translations are studied in ([BFL94]). They are called structural there, because more will be preserved of the structure of the formula when the formula is factored. We need a variation of the structural translation of ([BFL94]) to decide the guarded fragments.

Definition 2.5 Let $C = \{F_1, \ldots, F_n\}$ be a set of formulae. We define $\mathrm{Struct}(C)$ as the following operation:

- Put $D = \{\}$.

- If one of the F_i is a literal, then remove F_i from C and add F_i to D.
- If one of the F_i is not a literal, then write $F_i = F_i[A]$, where A is a minimal subformula of F_i that is not a literal, possibly the whole of F_i. Let $x_1, \ldots, x_n$ be an enumeration of the variables free in A. Let α be a predicate name, not occurring in C or D. Then add

$$\forall x_1, \ldots, x_n[(\neg\, \alpha(x_1, \ldots, x_n) \vee A) \wedge (\alpha(x_1, \ldots, x_n) \vee \neg\, A)]$$

  to D. Replace $F_i[A]$ by $F_i[\alpha(x_1, \ldots, x_n)]$ in C.
- If $C = \{\ \}$, then D is the result.

If C is in negation normal form, then Struct_+ is defined by adding $\forall x_1, \ldots, x_n[\neg\, \alpha(x_1, \ldots, x_n) \vee A]$ to D, instead of using equivalence. Struct_+ has the advantage that it is polarity preserving.

If C is in negation normal form, then $\text{Struct}_\forall$ is defined as follows:

- Put $D = \{\}$.
- If there is no occurrence of the $\forall$-quantifier in F_i, then remove F_i from C, and add F_i to D.
- If there is an occurrence of the $\forall$-quantifier in an F_i in C, then write $F_i = F_i[\forall y A]$, where A is a subformula of F_i, in which there is no $\forall$-quantifier. Let $x_1, \ldots, x_n$ be the variables that are free in $\forall y A$. Let α be a predicate name that does not occur in C or D. Then add

$$\forall x_1, \ldots, x_n, y[\neg\, \alpha(x_1, \ldots, x_n) \vee A] \text{ to } D.$$

  Replace $F_i[\forall y A]$ by $F_i[\alpha(\alpha_1, \ldots, \alpha_n)]$ in C.
- If $C = \{\ \}$, then D is the result.

Now we define Skolemization. Skolemization is the replacement of existential quantifiers by fresh function symbols in the preceding universal quantifiers.

Definition 2.6 Let $C = \{F_1, \ldots, F_n\}$ be a set of formulae in NNF. We define two types of *Skolemization*:

1. $\text{Sk}(C)$ is defined by making the following replacements as long as possible: As long as one of the F_i contains an existential quantifier, write $F_i = F_i[\exists y A]$, where $\exists y A$ is not in the scope of another existential quantifier. Let $x_1, \ldots, x_n$ be the universally quantified variables, in the scope of which A occurs. Replace $F_i[\exists y A]$ by $F_i[A[y := f(x_1, \ldots, x_n)]\ ]$.

2. Skc(C) is defined as Sk(C), (so, working from outside to inside), but instead of taking all variables $x_1, \ldots, x_n$ take only those variables, that are free in $\exists y A$, and bound by a universal quantifier in F.

There are more sophisticated ways for Skolemization leading to more general Skolem terms, see ([OWBach95]), but they are of no use for obtaining decision procedures.

Definition 2.7 Let $C = \{F_1, \ldots, F_n\}$ be a set of formulae in NNF containing no existential quantifiers: The *clausification* of C, written as Cls(C) is the result of the following replacements

- Replace $A \vee (B \wedge C)$ by $(A \vee B) \wedge (A \vee C)$.
- Replace $(A \wedge B) \vee C$ by $(A \vee C) \wedge (B \vee C)$.
- Replace $\forall x A$ by $A[x := X]$, where X is a designated variable symbol not occurring in A.
- If one of the $F_i = A \wedge B$, then replace F_i by A and B.

So now we have collected the following operations on formula sets:

Na(F) Remove $\leftrightarrow$ and $\rightarrow$.

NNF(C) Bring C in negation normal form.

Red(C) Move quantifiers inward as much as possible.

Prenex(C) Move quantifiers outward as much as possible.

Struct(C) Replace nested subformulae by fresh atoms, and add equivalence definitions for the new atoms.

Struct$_+$(C) Replace nested subformulae by fresh atoms, but add implications instead of equivalences.

Struct$_\forall$(C) Only replace universally quantified subformulae by fresh atoms.

Sk(C) Replace every existentially quantified subformula of C by a fresh function symbol.

Skc(C) Skolemize C carefully, i.e. using only the variables that are used.

Cls(C) Factor C into a set of clauses.

There are quite some variations possible in how to transform a set of formulae into clause form. It has been proven in ([BL94]) and ([BFL94]) that small variations may cause large differences in the length of the minimal proof.

In ([CL73]) the following method is recommended:

$$\text{Na}; \text{NNF}; \text{Prenex}; \text{Sk}; \text{Cls}.$$

The disadvantage of this method is the use of the prenex normal form. If one moves a universal quantifier outward, existential quantifiers may come into its scope. This gives rise to unnecessarily complicated Skolem terms. It has been shown in ([BL94]) that the price may by larger than any fixed, finite iteration of exponentials. For example in $\forall x_1 \exists y_1 P(x_1, y_1) \vee \forall x_2 \exists y_2 P(x_2, y_2)$, one of y_1 or y_2 will be unnecesarily in the scope of both x_1 and x_2.
In ([BL94]) and ([BFL94]) the following method is considered the best:

$$\text{Na}; \text{NNF}; \text{Red}; \text{Struct}; \text{Sk}; \text{Cls},$$

In order to obtain decision procedures it is necessary to use other transformations. We will define them when we need them.

2.2 Covering Literals

Most resolution decidable classes rely on the notion of (weakly) covering literals. The guarded fragment, and the loosely guarded fragment are no exception.

Definition 2.8

- A literal A is *covering* if every functional subterm t of A contains all variables of A.
- A literal A is *weakly covering* if every functional, non-ground subterm t of A contains all variables of A.
- A literal is *flat* if it contains only variables as arguments.
- A literal is *weakly flat* if all its arguments are either ground, or a variable.

Covering and weakly covering literals are typically the result of skolemization, when the prefix ends in an existential quantifier. If an atom $a(\overline{x}, y)$ in the scope of $\forall \overline{x} y$ is skolemized the result equals $a(\overline{x}, f(\overline{x}))$, which is covering. If $a(\overline{x}, y)$ contains ground terms, then the result is weakly covering.
The main property of (weakly) covering literals is that they do not grow when they are unified. Before we prove this we prove a technical property.

Lemma 2.9 Let A and B be two weakly covering literals, such that $\text{Var}(A) \subseteq \text{Var}(B)$, and $\text{Vardepth}(A) \leq \text{Vardepth}(B)$. Let Θ be the mgu of A and B. Then $\text{Vardepth}(A) = \text{Vardepth}(B)$.

proof:
Suppose that Vardepth$(A) \neq$ Vardepth(B). Then Vardepth$(A) <$ Vardepth(B). For every substitution Θ, it must be the case that Vardepth$(A\Theta) <$ Vardepth$(B\Theta)$, because every variable of A occurs in the deepest term of B. But then A and B cannot be unified.
end of proof

The following states that when two weakly covering terms are unified, the result is not larger than the largest of them.

Theorem 2.10 Let A and B be weakly covering literals that have an mgu Θ. Let $C = A\Theta = B\Theta$. Then:

1. C is weakly covering,
2. Vardepth$(C) \leq$ Vardepth(A), or Vardepth$(C) \leq$ Vardepth(B).
3. Varnr$(C) \leq$ Varnr(A) or Varnr$(C) \leq$ Varnr(B).

proof:
If we can prove that for either A or B, there is no variable V such that the mgu Θ assigns a non-ground functional term to V, then we are done. We consider the following cases:

- Vardepth$(A) =$ Vardepth(B). Assume that the mgu Θ is constructed by the algorithm of Definition 1.7. We will show that for every variable X, the result $X\Theta$ is ground, or a variable.

  Suppose there were a minimal difference of the form (X, t) (or converse), where t is non-ground, and functional. t has to contain all variables of B, since B is weakly covering. The variable X cannot occur in B, because of the occurs check. Define $\Theta' = \{X := t\}$. We have Var$(B\Theta') \subseteq$ Var$(A\Theta')$, and Vardepth$(B\Theta') <$ Vardepth$(A\Theta')$. This contradicts Lemma 2.9. By following the iteration of Definition 1.7 it follows that the unifier Θ does not assign a non-ground functional term to any of the variables in A or B.

- Vardepth$(A) <$ Vardepth(B). Assume that the mgu Θ is constructed by the algorithm of Definition 1.7.

  Let (t_1, t_2) be a minimal difference based on a term with maximal vardepth in B. (t_1, t_2) must be of the form (X, t), where X does not occur in t, and t is non-ground, functional, and contains all variables of B. Put $\Theta' = \{X := t\}$.

  We now have Var$(B\Theta') \subseteq$ Var$(A\Theta')$ and Vardepth$(B\Theta') <$ Vardepth$(A\Theta')$, because B was not affected by Θ'. By Lemma 2.9 it follows that
  Vardepth$(A\Theta') =$ Vardepth$(B\Theta')$. Let Σ be the most general unifier of

$A\Theta'$ and $B\Theta'$. By the first case Σ does not assign a non-ground, functional term to any variable. From the fact that Θ' did not affect B, and $\Theta = \Theta' \cdot \Sigma$ it follows that Θ does not assign a non-ground, functional term to a variable in B.

- $\mathrm{Vardepth}(A) > \mathrm{Vardepth}(B)$. The situation is symmetric to the previous case.

end of proof

The following is necessary in order to show that when two clauses resolve, and they consist of weakly covering literals, that then the resolvent consists of weakly covering literals. Let $R_1\Theta \cup R_2\Theta$ be a resolvent of $\{A_1\} \cup R_1$ and $\{\neg\, A_2\} \cup R_2$. Theorem 2.10 states that $A_1\Theta$ is weakly covering, but we have proven nothing about the literals in $R_i\Theta$.

Theorem 2.11 Let A and B be literals and let Θ be a substitution such that

1. $\mathrm{Var}(A) \subseteq \mathrm{Var}(B)$,
2. A is weakly covering,
3. $B\Theta$ is weakly covering.

Then $A\Theta$ is weakly covering.

proof:
Let t_2 be a non-ground, functional subterm of $A\Theta$. Let V_2 be a variable of $A\Theta$. We have to show that V_2 occurs in t_2.
There must exist a variable V_1, that occurs in A, and such that $V_1\Theta$ contains V_2. (If V_2 was not introduced by Θ, then $V_1 = V_2$.) There are two possibilities:

1. There is a $t_1[W]$ in A, such that $t_1[W\Theta] = t_2$. In that case $t_1[W]$ is non-ground, and functional, so it contains V_1. Because of this $t_2 = t_1[W\Theta]$ contains $V_1\Theta = V_2$.
2. There is a variable W in A, such that t_2 is a subterm of $W\Theta$. (This case includes $t_2 = W\Theta$) Because W also occurs in B, the term $W\Theta = t_2$ occurs in $B\Theta$. Because $B\Theta$ is weakly covering, V_2 must occur in t_2.

end of proof
Usually ([FLTZ93]) this theorem is found with a 4th condition: B is weakly covering. We have shown that this condition can be dropped. That this results in a true generalization can be seen by putting: $A = p(X, Y), B = p(X, s(Y)), \Theta = \{Y := s(X, Y)\}$. B is not weakly covering, but $A\Theta$ is.
No more conditions can be dropped from Theorem 2.11, as can be seen from the following examples:

1. $A = p(X,Y),\ B = p(X),\ \Theta = \{X := s(X)\}$.
2. $A = p(X,s(Y)),\ B = p(X,Y),\ \Theta = \{\}$.
3. $A = p(X,Y),\ B = p(X,Y),\ \Theta = \{X := s(X)\}$.

The following states that the side literals can be bound in size by the literals resolved upon.

Lemma 2.12 If

1. $\mathrm{Var}(A) \subseteq \mathrm{Var}(B)$,
2. A is weakly covering,
3. B is weakly covering,
4. $\mathrm{Vardepth}(A) \leq \mathrm{Vardepth}(B)$,

Then: $\mathrm{Vardepth}(A\Theta) \leq \mathrm{Vardepth}(B\Theta)$.

There is another technical fact that we are going to need, namely that unification does not introduce new ground terms in weakly covering literals, unless the result is completely ground.

Lemma 2.13 Let $C = A\Theta = B\Theta$ be the most general unifier of two weakly covering literals. If C is not ground by itself, then every ground term of C occurs either in A or in B.

proof:
If either A or B is ground, then C will be ground, so let us consider the case where neither A, nor B is ground.
Assume that Θ is obtained by the algorithm of Definition 1.7. Then we can write

$$\Theta = \Sigma_1 \cdot \ldots \cdot \Sigma_n,$$

where each Σ_i is a simple substitution of the form $\{X := t\}$.
We show by induction that if both $A\Sigma_1 \cdots \Sigma_i$ and $B\Sigma_1 \cdots \Sigma_i$ are non-ground, then they do not contain new ground terms. This is clearly true for $i = 0$. Suppose it is true for some i. Then $\Sigma_{i+1} = \{X := t\}$, where t is a term occurring either in $A\Sigma_1 \cdots \Sigma_i$, or $B\Sigma_1 \cdots \Sigma_i$. Hence there is no ground term, not in A or B, in t.
This means that the only manner in which a new ground term can be obtained is when t is ground, and X was the last variable in a functional term. Now if variable X occurs somewhere in a functional term, then this term either contains all variables of $A\Sigma_1 \cdots \Sigma_i$, or of $B\Sigma_1 \cdots \Sigma_i$. (Consequence of the proof of Theorem 2.10) Hence if the last variable is substituted away, either $A\Sigma_1 \cdots \Sigma_i$, or $B\Sigma_1 \cdots \Sigma_i$ is ground. Then one of $A\Theta$ or $B\Theta$ is ground, and so they are both ground.
end of proof

3 Decision by Resolution

In this chapter we give the decision procedures for the guarded fragment and the loosely guarded fragment. The decision procedures are based on ordered resolution, as defined in Definition 1.9. It is very common to restrict the resolution rule by an ordering, but usually this is done with the purpose of improving efficiency in the case that a proof exists. However certain orderings can be used to enforce termination in the case that a proof does not exist. We will illustrate this from an example. Let C be some clause set, in which only one variable X is used, all literals contain this variable X, and there are no constant symbols in C. So $\{p(X), q(s(X,X),X)\}$ is allowed, but $\{p(s(X),0)\}$ not. Let $\sqsubset$ be an order on literals that is defined from: $A \sqsubset B$ if $\text{Vardepth}(A) < \text{Vardepth}(B)$. Then there are the following facts:

1. Every ordered resolvent or factor from C contains exactly one variable, and no constants. So every derivable clause can be renamed s.t. that it contains only the variable X.

2. If Θ is the mgu of two literals A and B, which each contain exactly one variable, and no constant symbol, then $A\Theta$ and $B\Theta$ are also such literals, and $\text{Vardepth}(A\Theta) = \text{Vardepth}(B\Theta)$ is equal to $\text{Vardepth}(A)$ or to $\text{Vardepth}(B)$.

3. If $A < B$, and $\Theta = \{X := t\}$ is a substitution, such that t contains exactly one variable and no constants, then then $\text{Vardepth}(A\Theta) < \text{Vardepth}(B\Theta)$.

As a consequence, the clauses cannot become deeper, cannot contain more than one variable, and no new symbols are introduced. Because the set of literals that can occur in the clauses is finite, the set of clauses is finite. Hence the order $\sqsubset$ enforces termination. If one can prove the completeness of resolution with $\sqsubset$, at least for this one-variable class, then one has a decision procedure. This is straightforward, since orders satisfying the condition $A \sqsubset B \Rightarrow A\Theta \sqsubset B\Theta$ are known to be complete. ([CL73]). The decidability proofs will have the same structure as this example.

3.1 The Guarded Fragment

We now develop the decision procedure for the guarded fragment. We begin by defining the guarded fragment for first order logic. The definition in ([ANB96]) is slightly more restrictive, because we allow constants in the formulae. In ([ANB96]) only variables are allowed as terms.

Definition 3.1 The *guarded fragment* of first order logic ($\mathcal{GF}$) is recursively built up as follows:

1. $\top$ and $\bot$ are in $\mathcal{GF}$.

2. If A is an atom, such that none of its arguments is functional, then $A \in \mathcal{GF}$.
3. If $A \in \mathcal{GF}$, then $\neg\, A \in \mathcal{GF}$.
4. If $A, B \in \mathcal{GF}$, then $A \vee B,\ A \wedge B,\ A \rightarrow B,\ A \leftrightarrow B \in \mathcal{GF}$.
5. If $A \in \mathcal{GF}$ and a is an atom, for which
   (a) all arguments of a are non-functional,
   (b) every free variable of A is among the arguments of a,
   then $\forall \overline{x}(a \rightarrow A) \in \mathcal{GF}$, for every sequence of variables $\overline{x}$.
6. If $A \in \mathcal{GF}$ and a is an atom, for which
   (a) every argument of a is a non-functional,
   (b) every free variable of A is among the arguments of a,
   then $\exists \overline{x}(a \wedge A) \in \mathcal{GF}$.

The atoms a are called *guards.*

Example 3.2 The following formulae are in the guarded fragment:

$$\forall x_1(a(x_1) \rightarrow b(x_1)),\ \ \exists x_1(a(x_1) \wedge b(x_1)).$$

The formula

$$\exists a\ \exists x\ (c(a,x) \wedge \forall yz[b(x,y,z) \rightarrow c(x,y,a) \wedge b(y,y,z)])$$

is not guarded, because the second occurrence of a is not guarded by the $b(x,y,z)$. The formula

$$\exists x(c(a,x) \wedge \forall yz[b(x,a,y,z) \rightarrow c(x,y,a) \wedge b(y,y,z)])$$

is guarded.
The following formula, which expresses transitivity is not guarded:

$$\forall x_1x_2x_3[a(x_1,x_2) \rightarrow a(x_2,x_3) \rightarrow a(x_1,x_3)].$$

The modal formula $\Diamond(\Box a \wedge b)$ translates into

$$\exists x[R(c,x) \wedge (\ \forall y R(x,y) \rightarrow a(y)\ ) \wedge b(y)],$$

which is in the guarded fragment, as was promised in the introduction. (c is the present world).

In order to be able to do something with resolution we need a notion of guardedness for clause sets, and a way to translate guarded, first order formulae into guarded clause sets. The translation is not completely standard. Standard translations would transform guarded formulae into non-guarded clauses.
The first step of the transformation is the transformation into NNF. This can be done without problems:

- Rule 5 has to be replaced by: If $A \in \mathcal{GF}$, and a is an atom for which
  - all arguments of a are non-functional, and
  - every free variable of A is among the arguments of a,

  then $\forall \overline{x}(\neg\, a \vee A) \in \mathcal{GF}$, for every sequence of variables $\overline{x}$.
- Condition 6 can be completely dropped.

For clause sets then we define the following normal form:

Definition 3.3 A clause set C is called *guarded* if its clauses are guarded.
A clause c is called guarded if it satisfies the following conditions:

1. All literals $A \in c$ are weakly covering.
2. If c is not ground, then there is a literal $A \in c$ with $\mathrm{Vardepth}(A) = 0$, such that $\mathrm{Var}(A) = \mathrm{Var}(c)$, and A is negative.
3. If $\mathrm{Vardepth}(A) > 1$, then $\mathrm{Var}(A) = \mathrm{Var}(c)$.

The negative literal of Condition 2 is the guard. $\mathrm{Vardepth}(A) > 1$ means that A has a non-ground functional term. It follows from the definition that every ground clause is guarded. We give a few examples:

Example 3.4 Clause $\{p(0, s(0)), q(s(s0))\}$ is guarded because it is ground. The clause $\{\neg\, p(X), \neg q(X, Y), r(f(X, Y))\}$ is guarded by $\neg\, q(X, Y)$. The clause $\{\neg\, p(X), \neg q(Y), r(f(X, Y))\}$ is not guarded. Adding a literal $\neg\, a(X, Y, X, X, Y)$ would make the clause guarded.

Now we can continue with the translation: We need a modification of $\mathrm{Struct}_\forall$. $\mathrm{Struct}_\forall$ itself does not work because iterated $\forall$-quantifiers may share one guard.

Definition 3.5 Let $F = \{F_1, \ldots, F_n\}$ be a set of formulae in negation normal form. $\mathrm{Struct}_{\mathcal{GF}}(F)$ is obtained as follows:

- Put $D = \{\ \}$.
- If there is no quantifier $\forall \overline{x}(\neg\, a \vee A)$ in F_i, then remove F_i from F and add F_i to D.
- If there is one then write $F_i = F_i[\forall \overline{x}(\neg\, a \vee A)]$, where A contains no univeral quantifiers.

  Let $\overline{y}$ be the variables that occur in a, but not in $\overline{x}$.

  Let α be a predicate name that does not occur in F, nor in D. Add
  $$\forall \overline{xy}(\neg\, a \vee \neg\, \alpha(\overline{y}) \vee A)$$
  to D. Replace $F_i[\forall \overline{x}(\neg\, a \vee A)]$ by $F_i[\alpha(\overline{y})]$ in F_i.
- If $F = \{\ \}$, then D equals the result.

Theorem 3.6 Let $F \in \mathcal{GF}$.

1. $F' = (\text{Na}; \text{NNF})(F) \in \mathcal{GF}$. (using the modification for the negation normal form).
2. $F'' = \text{Struct}_{\mathcal{GF}}(F') \in \mathcal{GF}$.
3. $(\text{Sk}; \text{Cls})(F'')$ is a guarded clause set.

proof:
We study the steps made in the transformation: Na and NNF are characterized by a set of rewrite-rules. Let $\Phi = \forall \overline{x}(a \rightarrow A)$ or $\Phi = \exists \overline{x}(a \wedge A)$ be a guarded quantification. Φ will remain guarded under any rewriting inside A, since none of the rewrite rules introduces a free variable. Similarly if Φ occurs in the X or Y of a rewrite rule $(X \text{ op } Y) \Rightarrow \cdots$ then A is just copied without problems. The only possible problem is when $\forall \overline{x}(a \rightarrow A)$ rewrites to $\forall \overline{x}(\neg\, a \vee A)$, but for this case we extended the definition of the guarded fragment.
The next step is $\text{Struct}_{\mathcal{GF}}$. The formula $\forall \overline{xy}[\neg\, a \vee \neg\, \alpha(\overline{y}) \vee A]$ is guarded, since a is a guard, and A is not affected. Any quantification in which $\forall \overline{x}(\neg\, a \vee A)$ occurs remains guarded, when it is replaced by $\alpha(\overline{y})$, because no new free variables are introduced.
After this every existential quantifier occurs either outside the scope of any $\forall$-quantifier, in that case it will be replaced by a constant, or in the A of a guarded formula $\forall \overline{x}(\neg\, a \vee A)$. In the latter case it will be replaced by a function $f(\overline{x})$. The result is a formula in which all universal quantifiers are guarded, and all functions are Skolem functions. They are either constants or contain all variables of the guarded quantification in which they occur.

After that the formulae $\forall \overline{x}(\neg\, a \vee A)$ can be factored into guarded clauses

$$\forall \overline{x}(\neg\, a \vee A_1), \ldots, \forall \overline{x}(\neg\, a \vee A_n).$$

end of proof

Example 3.7 The guarded formula

$$\exists x\ n(x) \wedge \forall y[a(x,y) \rightarrow \neg\, \exists z < p(x,z) \wedge (\forall x\ a(x,z) \rightarrow (b(z,z) \wedge c(x,x))) >]$$

is translated as follows: First $(\text{Na}; \text{NNF})$ results in

$$\exists x\ n(x) \wedge \forall y[\neg\, a(x,y) \vee \forall z < \neg\, p(x,z) \vee (\exists x\ a(x,z) \wedge (\neg\, b(z,z) \vee \neg\, c(x,x))\ ) >].$$

After that $\text{Struct}_{\mathcal{GF}}$ results in the following set of formulae:

$$\exists x[\ n(x) \wedge \alpha(x)].$$

$$\forall xy[\neg\, a(x,y) \vee \neg\, \alpha(x) \vee \beta(x)].$$

$$\forall xz[\neg\, p(x,z) \vee \neg\, \beta(x) \vee (\exists x\ a(x,z) \wedge (\neg\, b(z,z) \vee \neg\, c(x,x)))].$$

Sk results in:

$$n(c) \wedge \alpha(c).$$

$$\forall xy[\neg\, a(x,y) \vee \neg\, \alpha(x) \vee \beta(x)].$$

$$\forall xz[\neg\, p(x,z) \vee \neg\, \beta(x) \vee (a(f(x,z),z) \wedge (\neg\, b(z,z) \vee \neg\, c(f(x,z),f(x,z))\ ))].$$

Clausification results in:

$\{n(c)\}$
$\{\alpha(c)\}$
$\{\neg\, a(X,Y), \neg\, \alpha(X), \beta(X)\}$
$\{\neg\, p(X,Z), \neg\, \beta(X), a(f(X,Z),Z)\}$
$\{\neg\, p(X,Z), \neg\, \beta(X), \neg\, b(Z,Z), \neg\, c(f(X,Z),f(X,Z))\}$.

We now will define the order on literals. We will be using completely standard ordered resolution with the only difference that the order is non-standard.

Definition 3.8 We define the following order $\sqsubset$ on literals:

1. $A \sqsubset B$ if $\mathrm{Vardepth}(A) < \mathrm{Vardepth}(B)$, or
2. $A \sqsubset B$ if $\mathrm{Var}(A) \subset \mathrm{Var}(B)$.

Lemma 3.9 Every guarded clause has a $\sqsubset$-maximal literal, and this literal contains all variables in the clause.

proof:
If the clause has literals with a non-ground functional term, then let A be a literal with maximal Vardepth. A is maximal, and A contains all literals in the clause.
If no literal in c has a non-ground functional term, then the guard is maximal, because it contains all literals in the clause.
end of proof

We will prove that resolution and factoring, restricted by $\sqsubset$ derives only a finite set of clauses from a guarded clause set.

Lemma 3.10 Let C be a finite set of guarded clauses. Let $\overline{C}$ be the set of clauses that can be derived from C by $\sqsubset$-ordered resolution, and by unrestricted factorization. Then:

1. Every clause in $\overline{C}$ is guarded.
2. $\mathrm{Varnr}(\overline{C}) \leq \mathrm{Varnr}(C)$.
3. $\mathrm{Vardepth}(\overline{C}) \leq \mathrm{Vardepth}(C)$.

proof:
There are 3 types of clauses in $\overline{C}$, namely clauses from C, clauses derived by resolution, and clauses derived by factorization.

- For initial clauses from C, the situation is trivial.
- Let c be obtained from two guarded clauses c_1 and c_2 by resolution. We show that

  1. c is guarded,
  2. $\mathrm{Varnr}(c) \leq \mathrm{Varnr}(c_1)$ or $\mathrm{Varnr}(c) \leq \mathrm{Varnr}(c_2)$, and that
  3. $\mathrm{Vardepth}(c) \leq \mathrm{Vardepth}(c_1)$ or $\mathrm{Vardepth}(c) \leq \mathrm{Vardepth}(c_2)$.

  Write $c_1 = \{A_1\} \cup R_1,\ c_2 = \{A_2\} \cup R_2$, where A_1 and A_2 are the complementary literals resolved upon. Let Θ be the mgu that was used. There are the following cases:

  - Both c_1 and c_2 are ground. Then the result is immediate.
  - c_1 is ground, and c_2 is non-ground. Because $A_2\Theta = A_1\Theta = A_1$ is ground, $R_2\Theta$ must be ground, since $\mathrm{Var}(R_2) \subseteq \mathrm{Var}(A_2)$. Because of this c must be ground. Then c is guarded, $\mathrm{Vardepth}(c) = -1$, and $\mathrm{Varnr}(c) = 0$.
  - c_2 is non-ground, c_1 is ground. This goes the same as the previous case.
  - Both c_1 and c_2 are non-ground.
    Let d be the maximum of $\mathrm{Vardepth}(c_1)$ and $\mathrm{Vardepth}(c_2)$. Let n be the maximum of $\mathrm{Varnr}(c_1)$ and $\mathrm{Varnr}(c_2)$. By Theorem 2.10, $\mathrm{Vardepth}(A_1\Theta) = \mathrm{Vardepth}(A_2\Theta) \leq d$, and $\mathrm{Varnr}(A_1\Theta) = \mathrm{Varnr}(A_2\Theta) \leq n$.
    Since $\mathrm{Vardepth}(R_i) \leq \mathrm{Vardepth}(A_i)$, and $\mathrm{Var}(R_i) \subseteq \mathrm{Var}(A_i)$, using Lemma 2.12, $\mathrm{Vardepth}(R_i\Theta) \leq \mathrm{Vardepth}(A_i\Theta)$ and $\mathrm{Var}(R_i\Theta) \subseteq \mathrm{Var}(A_i\Theta)$. This together ensures that $\mathrm{Vardepth}(R_1\Theta \cup R_2\Theta) \leq d$, and $\mathrm{Varnr}(R_1\Theta \cup R_2\Theta) \leq n$.
    It remains to show that c is guarded. Using Theorem 2.10 and Theorem 2.11 it follows that every literal in $R_i\Theta$ is weakly covering.
    Let $B \in c$, be with $\mathrm{Vardepth}(B) > 0$. We have to show that $\mathrm{Var}(B) = \mathrm{Var}(c)$. There must be an $B_i \in c_i$, with $B_i\Theta = B$. If B_i has non-ground, functional terms, then $\mathrm{Var}(B_i) = \mathrm{Var}(A_i)$, and the result follows from the fact that A_i contains all variables of c_i. Otherwise there is a variable X in B_i, for which $X\Theta$ is non-ground, and functional. Since X occurs in A_i, and $A_i\Theta$ is weakly covering, it must be the case that $\mathrm{Var}(X\Theta) = \mathrm{Var}(A_i\Theta)$. This ensures that $\mathrm{Var}(B_i\Theta) = \mathrm{Var}(A_i\Theta) = \mathrm{Var}(c)$.
    We still have to show that there is a negative literal $A \in c$, for which $\mathrm{Var}(A) = \mathrm{Var}(c)$. There are three cases:

1. Both A_1 and A_2 are not a guard. We can assume without loss of generality that Vardepth$(A_1) \geq$ Vardepth(A_2). The mgu Θ has the following property: If X is a variable of A_1, then $X\Theta$ is ground or a variable. Let G be a guard of c_1. Then $G\Theta$ is a guard of c, because $G\Theta$ is negative, has no non-ground, functional terms, and contains all variables of $A_1\Theta$. It is easy to see that $A_1\Theta$ contains all variables of c.
2. One of A_1 and A_2 is a guard, say A_2. Then Vardepth$(A_1) \geq$ Vardepth$(A_2$. Because of this the mgu has the same property: If X is a variable of A_1, then $X\Theta$ is ground or a variable. The rest is anologeous to the previous case.
3. Both of A_1 and A_2 are a guard. This cannot happen, because guards are negative. Fortunately.

- Let c be obtained from c_1 by factorization. We show that

  1. c is guarded,
  2. Varnr$(c) \leq$ Varnr(c_1), and
  3. Vardepth$(c) \leq$ Vardepth(c_1).

  If c is ground then the situation is trivial, otherwise we have $c_1 = \{A_1, A_2\} \cup R$, and $c = \{A_1\Theta\} \cup R\Theta$, where Θ is the mgu of A_1 and A_2.

  We show that for every variable X, the result $X\Theta$ is either a variable, or ground. If both A_1 and A_2 do not contain non-ground, functional terms then it is easily seen that the condition is satisfied. Otherwise let A_1 and A_2 be such that Vardepth$(A_1) \leq$ Vardepth(A_2). Certainly A_2 contains a non-ground functional term, and A_2 contains all variables of c_1. By Theorem 2.10 we see that Vardepth$(A_2\Theta) \leq$ Vardepth(A_2). This would not be true if an $X\Theta$ would be non-ground and functional. It is easily checked that c is guarded.

end of proof
There is a property that we did not prove, namely that guards are never introduced, and only copied. So the property of being guard is infusa, and not acquisita.
It remains to show that the set of clauses is finite.

Lemma 3.11 1. Let c be a non-ground factor of clause c_1. Clause c contains no ground terms, which are not in c_1.

2. Let c be a non-ground resolvent of clauses c_1 and c_2. Then c contains no ground terms, that are not in c_1 or c_2.

3. Let c be a resolvent of c_1 and c_2, where c_1 is ground, and c_2 is non-ground. Then Depth$(c) \leq$ Depth(c_1) or Depth$(c) \leq$ Depth(c_2).

4. Let c be a resolvent of c_1 and c_2, which are both ground. Then $\text{Depth}(c) \leq \text{Depth}(c_1)$, or $\text{Depth}(c) \leq \text{Depth}(c_2)$.

proof:

1. The proof of the 1st claim is analogeous to the proof of the 2nd claim.

2. Let A_1 in c_1, and A_2 in c_2 be the literals resolved upon. Let Θ be the mgu. It follows immediately from Lemma 2.13 that $A_1\Theta$ and $A_2\Theta$ do not contain new ground terms, but we have to worry a little bit about the other terms.

   For each variable V, the result $V\Theta$ does not contain new ground terms. (Since V is either unaffected, or it occurs in A_1 or A_2.) Let B be one of the other literals of c_1 or c_2. If $B\Theta$ would contain a new ground term, then a non-ground functional term t would have become ground by the substitution. It follows from the definiton of a guarded clause that t contained all variables of C. Hence $A_1\Theta$ and $A_2\Theta$ are ground, which contradicts the fact that c is non-ground.

3. Let C be a literal of c. If C originates from c_1, then there is no problem. Otherwise write $C = B\Theta$, where B is non-ground and occurs in c_2. Let A_1 and A_2 be the literals resolved upon. If $\text{Depth}(B\Theta) \leq \text{Depth}(B)$, then there is no problem. Assume that $\text{Depth}(B\Theta) > \text{Depth}(B)$. The deepest term in $B\Theta$ must have been created by the unifier Θ. Since $\text{Vardepth}(A_2) \geq \text{Vardepth}(B)$ and $\text{Var}(B) \subseteq \text{Var}(A_2)$, it must be the case that the same term has been created in $A_2\Theta$, at a position which is at least as deep. It follows that $\text{Depth}(B\Theta) \leq \text{Depth}(A_2\Theta)$. Now $A_2\Theta$ equals A_1, which occurs in c_2, because c_2 is ground.

4. Because every literal in c occurs either in c_1, or in c_2.

end of proof

Lemma 3.12 Let C be a finite set of guarded clauses. Let $\overline{C}$ be its closure under $\sqsubset$-ordered resolution, and factoring (unrestricted). Then $\overline{C}$ is finite in size.

proof:
The problem that we have now is that, although $\text{Vardepth}(\overline{C}) \leq \text{Vardepth}(C)$, and $\text{Varnr}(\overline{C}) \leq \text{Vardepth}(C)$, we have no upper bound for the ground terms. Let C_{ng} be the set of non-ground clauses in C. Let $\overline{C}_{ng}$ be the set of non-ground clauses that can be derived from C_{ng}. (So, $\overline{C}_{ng}$ is the total set of non-ground clauses that can be derived)

It follows from Lemma 3.11 that $\overline{C}_{ng}$ does not contain a ground term that is not in C_{ng}. Hence $\overline{C}_{ng}$ is finite in size.
After that $\overline{C}$ can be obtained from $\overline{C}_{ng}$ by deriving ground clauses. It follows from Lemma 3.11, that $\overline{C}$ is finite in size.
end of proof

If we prove completeness then we are ready: The situation is more difficult than in proving the completeness of the E^+ class. There the order is almost liftable with as only exception the ground case. Here the order is even more non-liftable, since there are also non-ground exceptions. For example we have:

1. $p(s(0), X) \sqsubset p(0, s(X))$ and $p(X, 0) \sqsubset p(s(X), s(0))$. The substitution $\{X := 0\}$ results in a conflict.
2. Also $\neg\, p(X, X) \sqsubset \neg\, q(X, Y)$ and $\neg\, q(X, X) \sqsubset \neg\, p(X, Y)$. The substitution $\{X := Y\}$ results in a conflict.

In the second case the conflict occurs in non-ground literals. The first type of conflict could be avoided by redefining the order $\sqsubset$. For example $A \sqsubset B$ if all the arguments of A occur as arguments in B, but inside a functional term. This idea was used by Zamov in ([FLTZ93]). This would be sufficient for the clauses that result from clausification of formulae in $\mathcal{GF}$. However the 2nd case cannot be avoided, so there is no way for deciding the guarded fragment by a liftable order.
The completeness proof is based on the resolution game. We need some technical preparation.

Definition 3.13 We assume that there is a fixed enumeration of the set of variables $\{X_0, X_1, X_2, \ldots\}$. A literal A is *normal* if variable X_{i+1} occurs only after an occurrence of variable X_i. (When the literal is written in the standard notation). Every literal A can be renamed into exactly one normal literal, which we call the *normalization* of A. We write $\overline{A}$ for the normalization of A.

So literal $p(X_0, X_1, X_2)$ is normal. Its renamings $p(X_1, X_0, X_2)$, and $p(X_1, X_2, X_3)$ are not normal. If two literals are renamings of each other, they have the same normalization.

Definition 3.14 Let $\Theta = \{V_1 := t_1, \ldots, V_n := t_n\}$ be a substitution. The complexity of Θ, written as $\#\Theta$ equals $\#t_1 + \cdots + \#t_n$.

Theorem 3.15 Resolution, using $\sqsubset$ is complete for guarded clause sets.

proof:
Let C be an unsatisfiable guarded clause set. Let $\overline{C}$ be a the set of clauses that can be obtained from C using $\sqsubset$-ordered resolution, and $\sqsubset$-ordered factoring. We show that $\overline{C}$ must contain the empty clause.

Write $C = \{c_1, \ldots, c_n\}$. Let

$$\Theta_{1,1}, \ldots, \Theta_{1,l_1},$$

$$\ldots,$$

$$\Theta_{n,1}, \ldots, \Theta_{n,l_n}$$

be a list of substitutions such that the set of clauses

$$c_1\Theta_{1,1}, \ldots, c_1\Theta_{1,l_1}, \ldots, c_n\Theta_{n,1}, \ldots, c_n\Theta_{n,l_n}$$

is propositionally unsatisfiable. Each $l_i \geq 0$.
First we annotate each clause in the Herbrand set with its representing clause set as follows: For each $c_i = \{A_1, \ldots, A_p\}$ and substitution $\Theta_{i,j}$, the set C_{hb} contains the clause

$$\{A_1\Theta_{i,j}{:}A_1, \ldots, A_p\Theta_{i,j}{:}A_p\}.$$

We call the literals of the form $a{:}A$ *indexed literals*. Extend the order $\sqsubset$ to indexed literals by $a{:}A \sqsubset b{:}B$ iff $A \sqsubset B$. Then define the following resolution and factoring rule for indexed clause sets:

resolution From $\{a{:}A_1\} \cup R_1$ and $\{\neg\, a{:}A_2\} \cup R_2$ derive $R_1\Theta \cup R_2\Theta$.

factoring From $\{a{:}A_1, a{:}A_2\} \cup R$ derive $\{a{:}A_1\Theta\} \cup R\Theta$.

In both cases Θ is the mgu. The result of a Θ on an indexed literal $b{:}B$ is defined as $b{:}(B\Theta)$. The literals resolved upon, and one of the literals factored upon must be maximal.
Given this resolution rule, let $\overline{C}_{hb}$ be the closure of C_{hb}. It is clear that if we can prove that $\overline{C}_{hb}$ contains the empty clause, then $\overline{C}$ contains the empty clause.
In order to do this define the following resolution game $\mathcal{G} = (P, \mathcal{A}, \prec)$, and initial clause set $C_{\mathcal{G}}$:

- The set P of literals equals the set of literals that occur as a in the $a{:}A$ of C_{hb}.
- The set $\mathcal{A}$ of attributes is constructed as follows: Let m be the maximal $\#\Theta_{i,j}$. Let $\mathcal{L}$ be the set of literals B for which there is a pair $a{:}A$ in one of the $\overline{C}_{hb}$, such that B is an instance of A, and a is an instance of B. Then $\mathcal{A}$ consists of the pairs (i, C), for which $0 \leq i \leq m$, and C is the normalization of a literal in $\mathcal{L}$. The set of attributes is finite in size.
- The order $\prec$ is defined from: $a_1{:}(i_1, C_1) \prec a_2{:}(i_2, C_2)$ if
    1. $i_1 < i_2$, or
    2. $i_1 = i_2$ and $(\mathrm{Varnr}(C_1) < \mathrm{Varnr}(C_2)$ or $\mathrm{Vardepth}(C_1) < \mathrm{Vardepth}(C_2))$.

- The initial clause set $C_{\mathcal{G}}$ consists of the indexed clauses that can be obtained as follows: For each indexed clause $\{a_1{:}A_1, \ldots, a_p{:}A_p\}$ in C_{hb}, there is the following clause in $C_{\mathcal{G}}$:

$$\{a_1{:}(k, \overline{A}_1), \ldots, a_p{:}(k, \overline{A}_p)\}.$$

Here $k = \#\Theta$, where Θ is the smallest substitution that makes $a_i = A_i\Theta$. The $\overline{A}_1, \ldots, \overline{A}_p$ are the normalizations of the $A_1, \ldots, A_p$.

This completes the resolution game.
Now the set $\overline{C}_{\mathcal{G}}$ is defined as the set $C_{\mathcal{G}}$, but taking $\overline{C}_{hb}$ as a starting point instead of C_{hb}. If we can show that $\overline{C}_{\mathcal{G}}$ contains the empty clause then we are done, since this implies that $\overline{C}_{hb}$ contains the empty clause.
We show that $\overline{C}_{\mathcal{G}}$ contains the empty clause by showing that it is a saturation of $C_{\mathcal{G}}$, based on $\mathcal{G}$.

- Let $c_1 = \{a{:}(k_1, \overline{A}_1)\} \cup R_1$ and $c_2 = \{\neg\ a{:}(k_2, \overline{A}_2)\} \cup R_2$ be clauses in $\overline{C}_{\mathcal{G}}$, for which $a{:}(k_1, \overline{A}_1)$ and $\neg\ a{:}(k_2, \overline{A}_2)$ are maximal. Let $d_1 = \{a{:}A_1\} \cup S_1$ and $d_2 = \{\neg\ a{:}A_2\} \cup S_2$ be the clauses in $\overline{C}_{hb}$ from which c_1 and c_2 originate. Then $a{:}A_1$ and $\neg\ a{:}A_2$ must be maximal in d_1 and d_2. For if some literal in d_1 would be larger, the corresponding literal in c_1 would also be larger, in c_1, since all the k in the (k, B) of the indices are equal. The same is true for d_2. Because of this d_1 and d_2 have a resolvent d. We must show that the clause $c \in \overline{C}_{\mathcal{G}}$, resulting from d is a reduction of the resolvent of c_1 and c_2.

  If one of the literals $b{:}B$ from R_1 is replaced by $b{:}B\Sigma$, (where Σ) is the mgu, then $b{:}(k, B)$ can be replaced by $b{:}(k', B\Sigma)$, where $k' < k$. The same is true for the literals from R_2. This ensures that the result is a reduction.

- The situation in the case of factoring is analogeous.

end of proof

The order $\sqsubset$ as we have defined it here is very basic, and it could be refined further to improve the efficiency. Every order $\sqsubset' \supseteq \sqsubset$ can be used to decide the guarded fragment.

3.2 The Loosely Guarded Fragment

In this section we show that the loosely guarded fragment can also be decided by resolution. The loosely guarded fragment is a generalization of the guarded fragment, which has been introduced in ([vBenthem97]). The guards need not be one literal, but they may consist of a group of literals satisfying certain conditions.

Definition 3.16 The *loosely guarded fragment* of first order logic ($\mathcal{LGF}$) is recursively built up as follows:

1. $\top$ and $\perp$ are in $\mathcal{LGF}$.
2. If A is an atom, such that all its arguments are variables, then $A \in \mathcal{LGF}$.
3. If $A \in \mathcal{LGF}$, then $\neg\, A \in \mathcal{LGF}$.
4. If $A, B \in \mathcal{LGF}$, then $A \vee B,\ A \wedge B,\ A \rightarrow B,\ A \leftrightarrow B \in \mathcal{LGF}$.
5. If $A \in \mathcal{LGF}$, $\overline{x}$ is a list of variables, and $a_1, \ldots, a_n$ is a list of atoms for which
   (a) every argument of each a_i is a variable,
   (b) every free variable of A, occurs together with every x_i of $\overline{x}$ in one of the a_k,

   then $\forall\overline{x}(a_1 \wedge \cdots \wedge a_n \rightarrow A) \in \mathcal{LGF}$.
6. If $A \in \mathcal{LGF}$, $\overline{x}$ is a list of variables, and $a_1, \ldots, a_n$ is a list of atoms for which
   (a) every argument of each a_i is a variable,
   (b) every free variable of A occurs together with every x_i of $\overline{x}$ in one of the a_k,

   then $\exists\overline{x}(a_1 \wedge \cdots \wedge a_n \wedge A) \in \mathcal{LGF}$.

The translation to clausal normal form is the same as the translation for the guarded fragment. Like was the case for the guarded fragment, the guards are only necessary for the positively occurring $\forall$-quantifiers, and for the negatively occurring $\exists$-quantifiers.

Definition 3.17 A clause set is called loosely guarded if its clauses are loosely guarded. A clause c is loosely guarded if it satisfies the following condition:

1. All literals in c are weakly covering.
2. If $\mathrm{Vardepth}(A) > 0$, then $\mathrm{Var}(A) = \mathrm{Var}(c)$.
3. If c is non-ground there is a group of negative literals $\neg\, A_1, \ldots, \neg\, A_p \in c$, such all $\mathrm{Vardepth}(A_i) = 0$, and for every pair X, Y of variables in c, there is an A_i that contains them both.

The A_i are the loose guard.

Theorem 3.18 Loosely guarded formulae can be translated into loosely guarded clause sets, using the following transformation:

1. $F' = (\mathrm{Na}; \mathrm{NNF})(F)$.

2. $F'' = (\mathrm{Struct}_{\mathcal{LGF}})(F')$.

3. $C = (\mathrm{Sk};\mathrm{Cls})(F'')$.

($\mathrm{Struct}_{\mathcal{LGF}}$ is the obvious modification of $\mathrm{Struct}_{\mathcal{GF}}$.)

proof:
The proof is analogeous to the proof of Theorem 3.6. However there is one interesting part concerning $\mathrm{Struct}_{\mathcal{LGF}}$. When $\mathrm{Struct}_{\mathcal{LGF}}$ replaces a loosely guarded quantification $\forall\overline{x}(\neg\, a_1 \vee \cdots \neg\, a_n \vee A)$ by $\alpha(\overline{y})$ and $\forall\overline{xy}(\neg\, a_1 \vee \cdots \neg\, a_n \vee \neg\, \alpha(\overline{y}) \vee A)$, then the loose guard $\neg\, a_1 \vee \cdots \neg\, a_n \vee \alpha(\overline{y})$ satisfies the following condition: For every pair of variables v_1, v_2 from $\overline{xy}$ there is a literal in which they occur together. For if either v_1 or v_2 is among the $\overline{x}$, then v_1 and v_2 occur together in one of the $\neg\, a_i$, because the original quantification was loosely guarded. If both v_1 and v_2 are not among the $\overline{x}$, then they are both among the $\overline{y}$, and they occur together in $\neg\, \alpha(\overline{y})$.
end of proof

The decision procedure for loosely guarded clause sets is much more complicated than the decision procedure for guarded clause sets. The obvious approach is to use the closest approximation of the strategy for the guarded fragment, i.e. to first prefer the literals with maximal Vardepth, and after that the loose guard. When the loose guard is preferred hyperresolution has to be used instead of ordinary resolution. At this point unfortunately growth of Vardepth is possible, as is clear from the following example:

Example 3.19 The following clause is loosely guarded:

$$c_0 = \{\neg\, a_1(X,Y), \neg\, a_2(Y,Z), \neg\, a_3(Z,X),\ b_1(X,Y), b_2(Y,Z), b_3(Z,X)\}.$$

There is a hyperresolvent possible with the following clauses:

$$\begin{aligned} c_1 &= \{\neg\, p_1(A), a_1(s(A), s(A))\}, \\ c_2 &= \{\neg\, p_2(B), a_2(B, t(B))\}, \\ c_3 &= \{\neg\, p_3(C), a_3(t(C), C)\}. \end{aligned}$$

and substitution $\Theta = \{X, Y, B, C := s(A), Z := t(s(A))\}$. The result equals

$$\{\neg\, p_1(A), \neg\, p_2(s(A)), \neg\, p_3(s(A)), b_1(s(A), s(A)), b_2(s(A), t(s(A)), b_3(t(s(A)), s(A))\}$$

which has a Vardepth of 2, which is larger than we would like to see.

The explanation for this problem is as follows: Clause c_0 can hyperresolve with clause c_2 and c_3 with substitution $\Theta = \{Y, B, X := C, Z := t(C)\}$. The result equals:

$$\{\neg\, a_1(C,C), \neg\, p_2(C), \neg\, p_3(C), b_1(C,C), b_2(C, t(C)), b_3(t(C), C))\},$$

which is loosely guarded, and not too deep. In order to obtain the final hyperresolvent one needs to resolve upon the literal $\neg\, a_1(C, C)$. However $a_1(C, C)$ is not the deepest term in the clause, and when $a_1(C, C)$ is unified with $a_1(s(A), s(A))$ the literal $b_2(c, t(C))$ grows to a Vardepth of two.
What we will do is instead of generating a full hyperresolvent, generate a partial hyperresolvent that is not too large. We will prove that when a hyperresolvent is possible, using the guard, that then there is always a partial hyperresolvent possible which does not grow in Vardepth.

Theorem 3.20 Let literals $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$ together with substitution Θ satisfy the following conditions:

1. All A_i have $\mathrm{Vardepth}(A_i) = 0$.
2. For all $X, Y \in \mathrm{Var}(A_1, \ldots, A_n)$, there is an A_i such that $X, Y \in \mathrm{Var}(A_i)$.
3. All B_i are weakly covering, and have $\mathrm{Vardepth}(B_i) > 0$.
4. $i \neq j \Rightarrow \mathrm{Var}(B_i) \cap \mathrm{Var}(B_j) = \emptyset$.
5. Always $\mathrm{Var}(A_i) \cap \mathrm{Var}(B_j) = \emptyset$.
6. Θ is the mgu of $(A_1, B_1), \ldots, (A_n, B_n)$

Then it is possible to permute the A_i, B_i, and to find an $m \leq n$, such that, when Θ' is the mgu of $(A_1, B_1), \ldots, (A_m, B_m)$,

1. $\mathrm{Varnr}(B_1\Theta') \leq \mathrm{Varnr}(B_1)$, and $\mathrm{Vardepth}(B_1\Theta') \leq \mathrm{Vardepth}(B_1)$.
2. $\mathrm{Var}(B_i\Theta') \subseteq \mathrm{Var}(B_1\Theta')$, and $\mathrm{Vardepth}(B_i\Theta') \leq \mathrm{Vardepth}(B_1\Theta')$.
3. $\mathrm{Var}(A_i\Theta') = \mathrm{Var}(B_i\Theta')$, and $\mathrm{Vardepth}(A_i\Theta') = \mathrm{Vardepth}(B_i\Theta')$.

proof:
(3) follows obviously from the fact that Θ' is an mgu. Before we continue we need a notion: When a variable V occurs as $A_i(\ldots, V, \ldots)$, and a term t as $B_i(\ldots, t, \ldots)$, we say that V is *paired* to t.
If for all variables in the A_i the result $A_i\Theta$ is ground, then the theorem follows trivially.
Otherwise define the following order those variables V occurring in the $(A_1, \ldots, A_n)$ for which $V\Theta$ is not ground:

$$X \sqsubset Y \text{ if}$$

X and Y occur together in an A_i, as $A_i(\ldots, X, \ldots, Y, \ldots)$, and in the corresponding B_i there is $B_i(\ldots, T, \ldots, U, \ldots)$, with $\mathrm{Vardepth}(T) < \mathrm{Vardepth}(U)$.
Then the following holds:

MAXVAR There exists a $\sqsubset$-maximal variable in $(A_1, \ldots, A_n)$

If there does not exist a maximal variable this is caused by the fact that there is a cycle as follows:

$$V_0 \sqsubset V_1 \sqsubset \cdots \sqsubset V_p \sqsubset V_0.$$

We show that in this case there does not exist a unifier. The cycle is caused by literals of the form:

$$A_0(V_0, V_1), A_1(V_1, V_2), A_2(V_2, V_3), \ldots, A_p(V_p, V_0),$$

and

$$B_0(t_0, u_0), B_1(t_1, u_1), B_2(t_2, u_2), \ldots, B_p(t_p, u_p),$$

with $\text{Vardepth}(t_i) < \text{Vardepth}(u_i)$. Because the t_i and u_i are weakly covering, $\text{Vardepth}(t_i\Theta) < \text{Vardepth}(u_i\Theta)$, ($t_i\Theta$ and $u_i\Theta$ need not be weakly covering, but that is not important) Because $u_i\Theta = V_{i+1}\Theta$, for $i < p$, and $u_p\Theta = t_0\Theta$ it follows that

$$\text{Vardepth}(t_0\Theta) < \text{Vardepth}(t_1\Theta) < \cdots < \text{Vardepth}(t_p\Theta) < \text{Vardepth}(t_0\Theta),$$

which is impossible.
We can now construct the permutation. Permute all the (A_i, B_i) where A_i contains the maximal variable Z before the other (A_i, B_i). Let m be the index of the last A_i that contains Z. The (A_i, B_i) have the following property:

MAXVARDEPTH For every pair (A_i, B_i), if Z is matched to a term t of B_i, then $\text{Vardepth}(t) = \text{Vardepth}(B_i)$.

Suppose there is a term u in B_i, for which $\text{Vardepth}(u) > \text{Vardepth}(t)$, and u is not matched to Z. There are two possibilities:

1. u is paired to another variable, which contradicts the $\sqsubset$-maximality of Z, or

2. u is paired to a ground term. This would make $u\Theta$ ground. Since $\text{Vardepth}(u) > 0$, it follows that u contains all variables of B_i. But then $B_i\Theta$ is ground, and this contradicts the fact that $Z\Theta$ is non-ground.

Permute a pair (A_i, B_i) for which $\text{Vardepth}(B_i)$ is maximal on the first position. Let Θ' be the mgu of the $(A_1, B_1), \ldots, (A_m, B_m)$.
We must show that the permutation, and Θ' have the desired properties. Write $\Theta' = \Sigma_1 \cdot \Sigma_2 \cdot \Sigma_3 \cdot \Sigma_4 \cdot \Sigma_5$.

1. Σ_1 is the substitution that makes ground all variables in the A_i that are paired to a ground term. Z is not among these variables. Then:

   (a) $\text{Vardepth}(B_i\Sigma_1) \leq \text{Vardepth}(B_i)$, and $\text{Varnr}(B_i\Sigma_1) \leq \text{Varnr}(B_i)$, because Σ_1 does not affect the B_i.

(b) $\text{Vardepth}(A_i\Sigma_1) \leq \text{Vardepth}(A_i)$, and $\text{Varnr}(A_i\Sigma) \leq \text{Varnr}(A_i)$, because variables are replaced by ground terms.

2. $\Sigma_2 = \{Z := t\}$, where t is a term of maximal Vardepth occurring in $B_1\Sigma_1$, and Z is a $\sqsubset$-maximal variable. It must be the case that $\text{Vardepth}(t) > 0$, $\text{Vardepth}(t) = \text{Vardepth}(B_1\Sigma_1) = \text{Vardepth}(B_1)$, and $\text{Vardepth}(B_1) > 0$ by assumption. Because of this t contains all variables of $B_1 = B_1\Sigma_1$. Σ_2 does not affect any of the $B_i\Sigma_1$, because Z occurs only in the A_i. We now have

   (a) $\text{Var}(B_1\Sigma_1\Sigma_2) \subseteq \text{Var}(A_i\Sigma_1\Sigma_2)$, because every $A_i\Sigma_1\Sigma_2$ contains t.

   (b) $\text{Vardepth}(A_i\Sigma_1\Sigma_2) = \text{Vardepth}(B_1)$, because t is the only non-ground and functional term in $A_i\Sigma_1\Sigma_2$.

   (c) $\text{Vardepth}(B_i\Sigma_1\Sigma_2) \leq \text{Vardepth}(A_i\Sigma_1\Sigma_2) = \text{Vardepth}(B_1)$, because $\text{Vardepth}(B_i\Sigma_1\Sigma_2) = \text{Vardepth}(B_i)$.

3. Σ_3 is the unifier of t with the remaining terms with t is paired. These are the terms with which Z was paired. Since they are weakly covering, and maximal in the B_i, we have the following:

   (a) $\text{Vardepth}(A_i\Sigma_1\Sigma_2\Sigma_3) \leq \text{Vardepth}(A_i\Sigma_1\Sigma_2)$. This follows from Theorem 2.10,

   (b) $\text{Vardepth}(B_i\Sigma_1\Sigma_2\Sigma_3) = \text{Vardepth}(t\Sigma_1\Sigma_2\Sigma_3)$. This follows from Theorem 2.10, and the fact that the terms with which t is paired are the terms with maximal Vardepth.

   (c) $\text{Var}(B_i\Sigma_1\Sigma_2\Sigma_3) \subseteq \text{Var}(B_1\Sigma_1\Sigma_2\Sigma_3)$.

4. Σ_4 is a substitution that replaces each of the remaining variables in the A_i by one of the terms with which it is paired. We have $\text{Var}(A_i\Sigma_1\Sigma_2\Sigma_3\Sigma_4) = \text{Var}(B_i\Sigma_1\Sigma_2\Sigma_3\Sigma_4)$ and $\text{Vardepth}(A_1\Sigma_1\Sigma_2\Sigma_3\Sigma_4) \leq \text{Vardepth}(B_1)$.

5. Σ_5 is the remaining unification. Since Σ_5 unifies terms with the same set of variables, Σ_5 must assign either a variable, or a ground term to each variable, hence the depth cannot increase.

The result follows by collecting all the inclusions and inequalities.
end of proof

This leads to the following resolution strategy:

Definition 3.21 The following constructions are allowed:

1. Let c be a clause. If c has a factor, then the construction of this factor is always allowed.

2. Let $c_1 = \{A_1\} \cup R_1$ and $c_2 = \{\neg\, A_2\} \cup R_2$ be clauses such that A_1 and A_2 are unifiable. Construction of the resolvent is allowed if for both i from $1, 2$ either c_i is ground, or (Vardepth$(c_i) > 0$, and Vardepth$(c_i) =$ Vardepth(A_i)).

3. Let $c = \{\neg\, A_1, \ldots, \neg\, A_p\} \cup R$ be non-ground and Vardepth$(c) = 0$. The A_i are the loose guard. If there are clauses $c_1 = \{A'_1\} \cup R_1, \ldots, c_p = \{A'_p\} \cup R_p$, s.t.

   (a) For each i, either c_i is ground, or (Vardepth$(c_i) > 0$ and Vardepth$(A'_i) =$ Vardepth(c_i)), and

   (b) a hyperresolvent is possible,

   then construct a permutation, and a selection of the A_i as in Theorem 3.20. Then construction of the following partial hyperresolvent is allowed: From $c = \{\neg\, A_1, \ldots, \neg\, A_m, \neg\, A_{m+1}, \ldots, \neg\, A_n\} \cup R$ and $\{A'_1\} \cup R_1, \ldots, \{A'_m\} \cup R_m$ construct $\{\neg\, A_{m+1}\Theta', \ldots, \neg\, A_n\Theta'\} \cup R\Theta' \cup R_1\Theta' \cup \cdots \cup R_m\Theta'$.

Once we have 3.20 the termination of the strategy is analogeous to Theorem 3.10. However the completeness proof is more complicated. The variation of hyperresolution that we use here is a completely new one, as far as the auther knows. It differs from the usual resolution strategies in the following main thing: The literals that can be used in a clause cannot be indicated in advance. It may be that one mixed clause has different preferences when it resolves with different clauses. In ([BG90]) resolution with selection functions is introduced, but there the usable literals are also selected in advance.

Theorem 3.22 Let C be a loosely guarded clause set. Let n be Varnr(C), and let $d =$ Vardepth(C). Then for every derivable clause c, we have Vardepth$(c) \leq d$, and Varnr$(c) \leq n$. Every derivable clause c is loosely guarded, and there is only a finite set of derivable clauses.

proof:

1. Suppose that c has been obtained by factorization from a c_1 by merging two literals with a substitution Θ. From Lemma 2.9, it follows that only literals of equal Vardepth can be unified. From Theorem 2.10 it follows that then each $V\Theta$ must be ground or a variable. Then $c_1\Theta$ has the desired properties.

2. Let c be obtained from c_1 and c_2 by binary resolution, using an mgu Θ. In the same manner as is the case with the guarded fragment, the $\sqsubset$-maximal literals contain all variables of their clause. If one of c_1, c_2 is ground, the result c is ground, and the desired properties follow immediately. Otherwise, w.l.o.g. assume that Vardepth$(c_1) \leq$ Vardepth(c_2). It

follows from Theorem 2.10 that for each variable V in c_2, the substitution $V\Theta$ is ground or a variable. This makes $c_2\Theta$ loosely guarded, and $c_2\Theta$ has the desired complexity bounds. Adding the literals of $c_1\Theta$ is harmless because $\text{Var}(c_1\Theta) \subseteq \text{Var}(c_2\Theta)$. (This ensures that c is loosely guarded, and $\text{Varnr}(c) \leq n$), Because the literal resolved upon in c_1 has maximal Vardepth, and contains all variables of c_1, it follows that $\text{Vardepth}(c) \leq n$.

3. Let c be obtained by incomplete hyperresolution from $\{\neg A_1, \ldots, \neg A_m\} \cup R$ and $c_1, \ldots, c_m$ with a substitution Θ. The argument goes pretty much the same as in the previous case, using Theorem 3.20 instead of 2.10. $c_1\Theta$ is loosely guarded, $\text{Varnr}(c_1\Theta) \leq n$, and $\text{Vardepth}(c_1\Theta) \leq d$. The other $c_i\Theta$ are controled by Condition 3 of Theorem 3.20, and $R\Theta$ by condition 2. This makes the result loosely guarded.

It remains to show that the set of clauses that can be derived is finite. Let G be the set of ground terms in C. G is clearly finite, because C is finite. The only manner in which a new ground term can be generated is in the generation of a ground clause, because of Lemma 2.13. Because of this the set of non-ground clauses is finite. Also the set of ground clauses with one non-ground parent is finite. After that the total set of clauses is finite.
end of proof

Definition 3.23 A *resolution game* $\mathcal{G} = (P, \mathcal{A}, \sqsubset)$ consists of:

- P : a set of literals.
- $\mathcal{A}$: a set of attributes.
- $\sqsubset$: a total and well-founded order on $P \times \mathcal{A}$.

Clauses are finite multisets of indexed literals. *Rules* are finite pairs of multisets of indexed literals of the form

$$[a_1{:}A_1, \ldots, a_p{:}A_p] \vdash [b_1{:}B_1, \ldots, a_q{:}A_q].$$

Reductions are as usual. c_2 is a reduction of c_1 if c_2 is obtained from c_1 by replacing zero or more indexed literals $a{:}A_1$ by $a{:}A_2$ with $a{:}A_2 \sqsubset a{:}A_1$, or by deleting them.

If a clause contains some negative indexed literals then it is possible to derive a rule from it as follows: From clause

$$[\neg\ a_1{:}A_1, \ldots, \neg\ a_p{:}A_p] \cup R$$

derive rule

$$[a_1{:}A_1, \ldots, a_p{:}A_p] \vdash R,$$

with $p > 0$. It is allowed that there are negative indexed literals among the R, and there exists more than one way to derive a rule from a clause.

We define an extension from $\sqsubset$ to clauses as follows: $c_1 \sqsubset c_2$ if the maximal indexed literal in c_1 is less than the maximal indexed literal of c_2, or the maximal indexed literal in c_1 is equal to the maximal indexed literal in c_2, but occurs less often.

We extend the order $\sqsubset$ to rules as follows: The position of $[a_1{:}A_1, \ldots, a_p{:}A_p] \vdash R$ in the order is determined by the position of $[\neg\, a_1{:}A_1, \ldots, \neg\, a_p{:}A_p] \cup R$.

Let C be a set of clauses of $\mathcal{G}$. A *saturation* of C is an ordered pair $(\overline{\mathcal{R}}, \overline{C})$, that satisfies the following conditions:

INITIAL For every clause in C, there is a reduction c', such that either $c' \in \overline{C}$, or there is a rule derived from c' in $\overline{\mathcal{R}}$.

FACTOR If $c_1 \in \overline{C}$ has form $[a{:}A_1, a{:}A_2] \cup R$, and $a{:}A_1$ is $\sqsubset$-maximal in c_1, then there is a reduction c' of $[a{:}A_1] \cup R$, such that either $c' \in \overline{C}$, or there is a rule derived from $c' \in \overline{\mathcal{R}}$.

BINARY If $c_1 = [a{:}A_1] \cup R_1 \in \overline{C}$, and $c_2 = [\neg\, a{:}A_2] \cup R_2 \in \overline{C}$, where $a{:}A_1$ is maximal in c_1, and $\neg\, a{:}A_2$ is maximal in c_2, then there is a reduction c' of $R_1 \cup R_2$, such that either $c' \in \overline{C}$, or there is a rule derived from c' in $\overline{\mathcal{R}}$.

HYPER If

$$r = [a_1{:}A_1, \ldots, a_p{:}A_p] \vdash R \in \overline{\mathcal{R}},$$

and there are clauses $[a_1{:}A_1'] \cup R_1, \ldots, [a_p{:}A_p'] \cup R_p \in \overline{C}$, such that all $a_i{:}A_i'$ are maximal in their clauses, then there is a permutation of the i, and a $q \leq p$ s.t. there is a reduction c' of

$$c = [\neg\, a_{q+1}{:}A_{q+1}, \ldots, \neg\, a_p{:}A_p] \cup R_1 \cup \cdots \cup R_q \cup R, \text{s.t.}$$

either $c' \in \overline{C}$ or a rule derived from c' in $\overline{\mathcal{R}}_2$. It must be the case that $c' \sqsubset r$.

Theorem 3.24 Let $(\overline{\mathcal{R}}, \overline{C})$ be a saturation of a clause set C. If C does not contain the empty clause, then C has a model.

proof:
The proof analogeous to the proof in ([GMW97]). We construct the following set I by induction on $\sqsubset$: Let k be the ordinal length of $\sqsubset$.

- $I_0 = \{ \}$.
- For each successor ordinal $\lambda + 1$, let $a{:}A$ be the indexed literal on position λ. Put $I_{\lambda+1} = I_\lambda \cup \{a{:}A\}$ if

1. there is a reduction of $a{:}A$ in I_λ, or
2. there is a clause $c \in \overline{C}$, which cannot be factored, such that $c \cap I_\lambda = \emptyset$, and $a{:}A$ is the maximal literal of c.

Otherwise put $I_{\lambda+1} = I_\lambda$.

- For each limit ordinal λ, put $I_\lambda = \bigcup_{\mu<\lambda} I_\mu$.

I_k satisfies the following property:

MAXUNIQUE For every indexed literal $a{:}A$ present in I_k there is a reduction $a{:}A'$ and a clause $c \in \overline{C}$ without factors, such that $a{:}A'$ is the maximal literal in c, and all other indexed literals of c are not in I_k.

Let $s = [a_1{:}A_1, \ldots, a_p{:}A_p]$ be a multiset of indexed literals. We say that I_k is *true* in I_k, if either

1. $s \cap I_k \neq [\,]$, or
2. s contains a negative indexed literal $\neg\, a{:}A$ for which there is no $a{:}A' \in I_k$. Note that $a{:}A'$ is not necessarily a reduction.

A rule $r = [a_1{:}A_1, \ldots, a_p{:}A_p] \vdash [b_q{:}B_q, \ldots, b_q{:}B_q]$ is true in I_k when the following holds:

1. If there are indices $A_1', \ldots, A_p'$, such that all $a_i'{:}A_i' \in I_k$, then $[b_1{:}B_1, \ldots, b_q{:}B_q]$ is true in I_k.

We will show that I_k has the following properties:

1. All clauses $c \in \overline{C}$ which do not have a factor, are true in I_k.
2. All rules $r \in \overline{\mathcal{R}}$ and all clauses $c \in \overline{C}$ that have a factor are true in I_k.
3. The rules $[a{:}A_1, \neg\, a{:}A_2] \vdash [\,]$ are true in I_k. (This is equivalent to saying that I_k does not contain a complementary pair)

A model can be easily extracted from this I_k, by interpreting $I(a) = t$ iff there is an $a{:}A \in I$.
It is evident from the construction of I_k, that all clauses $c \in \overline{C}$ without factors are true in I_k, since $c \cap I_k \neq \emptyset$.
Also the following holds: Let rule r be derived from clause c. If r is true in I_k, then c is true in I_k. Let $c = [\neg\, a_1{:}A_1, \ldots, \neg\, a_p{:}A_p] \cup R]$ be a clause. Let $r = [a{:}A_1, \ldots, a_p{:}A_p] \vdash R$ be derived from c. Assume that r is true in I_k. We proceed by cases:

1. If R is true in I_k, then clearly c is true in I_k.
2. Otherwise there must be an i, such that there is no A', for which $a_i{:}A' \in I_k$, but this also makes c true in I_k.

We now show by simultaneous induction on the $\sqsubset$-maximal element of r or c the following properties:

1. If $r \in \overline{\mathcal{R}}$ is a rule, then r is true in I_k.
2. If $c \in \overline{C}$ is a clause that has a factor, then c is true in I_k.

Assume $r = [a_1{:}A_1, \ldots, a_p{:}A_p] \vdash [b_1{:}B_1, \ldots, b_q{:}B_q]$ is a rule, and all smaller rules and clauses are true in I_k,
If there are A_i', such that all indexed literals $a_i{:}A_i'$ are in I_k, then by MAXUNIQUE there exist clauses $c_i = [a_i{:}A_i'] \cup R_i$, such that the $[a_i{:}A_i']$ are maximal in the c_i, and the c_i cannot be factored. By induction the resolvent

$$b_1{:}B_1, \ldots, b_q{:}B_q] \cup R_1 \cup \cdots \cup R_p$$

is true in I_k. Either it cannot be factored, or it is in $\overline{C}$, or there is a rule derived from it in $\overline{R}$. From this it follows that $[b_1{:}B_1, \ldots, b_q{:}B_q]$ is true in I_k.
Assume that $[a_1{:}A, a{:}A_2] \cup R$ is a clause which has a factor. This factor c must be true in I_k. It follows that c_1 is true in I_k.

It remains to show that there is no complementary pair in I_k. This is done completely analogeously to showing that the rules are true.
end of proof

Lemma 3.25 Let c be loosely guarded clause. Let Θ be a substitution, s.t. for each variable V, the result $V\Theta$ is either a variable, or a ground term. Then $c\Theta$ is loosely guarded.

proof:

1. Every literal in $c\Theta$ has form $A\Theta$, where A is weakly covering. Because Θ preserves the property of being weakly covering, $A\Theta$ is weakly covering.
2. If a literal $A\Theta$ has $\text{Vardepth}(A\Theta) > 0$, then $\text{Vardepth}(A) > 0$. Because A contained all variables of c, the result $A\Theta$ must contain all literals of $c\Theta$.
3. Let $V_1\Theta, V_2\Theta$ be a pair of variables occurring in $c\Theta$. There must be a negative literal $\neg\, A$ containg both V_1 and V_2. Then $\neg\, A\Theta$ is still very negative, and contains both $V_1\Theta$ and $V_2\Theta$.

end of proof

Theorem 3.26 The strategy of Definition 3.21 is complete for clause sets C in the loosely guarded fragment.

proof:
One we have the resolution game of Definition 3.23, the proof is analogeous to the proof of Theorem 3.15. Let C be an unsatisfiable, loosely guarded clause set. Let $\overline{C}$ be its closure under resolution and factoring. Let C_{hb} and $\overline{C}_{hb}$ be obtained as in Theorem 3.15.
The resolution game can be constructed in almost the same manner. When a clause $c \in \overline{C}$ has Vardepth$(c) > 0$, then the clause c' derived from it can be added to $\overline{C}_{\mathcal{G}}$. If Vardepth$(c) = 0$, then write $c = \{\neg\, a_1{:}A_1, \ldots, \neg\, a_n{:}A_n\} \cup R$, where the $\neg\, a_i{:}A_i$ are the loose guard, and add the clause $c' = [a_1{:}(k, \overline{A}_1), \ldots, a_n{:}(k, \overline{A}_n)] \vdash \overline{R}$ to $\overline{R}_{\mathcal{G}}$.
It can be shown that $(\overline{C}_{\mathcal{G}}, \overline{R}_{\mathcal{G}})$ is closed. When a rule r is used, the loose guard will always be instantiated. This ensures that the result is smaller than r.
end of proof

4 Conclusions & Further Work

We have shown that it is possible to effectively decide the guarded fragment and the weakly guarded fragment by resolution. The proofs that the resolution refinements are complete and terminating can be used as proofs for the decidability of these fragments, but they offer more than that. They also define practical decision procedures, using techniques that are standard to the theorem proving community.
Future work should be the comparison of the complexity of the procedures with the theoretical complexity results obtained in ([Graedel97]). Also some solution should be found for transitivity axioms. Transitivity axioms are non-guarded, and it has been shown in ([Graedel97]) that adding transitivity axioms leads to undecidability. Nevertheless there are modal logics ($S4$ and $K4$) that are based on transitive frames which are decidable. So it must be possible to combine some weaker version of the guarded fragment with transitivity.
Translation into the guarded fragment is not the optimal way of deciding simple modal logics like K and T, since these logics are in PSPACE, where the complexity of the guarded fragment is doubly exponential. Do there exist appropriate restrictions of the guarded fragment with lower complexity?

References

[ANB96] H. Andréka, J. van Benthem, I. Németi, Modal Languages and Bounded Fragments of Predicate Logic, ILLC Research Report ML-96-03, 1996.

[BFL94] M. Baaz, C. Fermüller, A. Leitsch, A Non-Elementary Speed Up in Proof Length by Structural Clause Form Transformation, In LICS 94.

[BL94] M. Baaz, A. Leitsch, On Skolemization and Proof Complexity, Fundamenta Informatica, Vol. 20-4, 1994.

[BG90] L. Bachmair, H. Ganzinger, On restrictions of Ordered Paramodulation with Simplification, In CADE 10, pp. 427-441, 1990.

[vBenthem96] J. van Benthem, Exploring Logical Dynamics, CSLI Publications, Stanford, California USA, 1996.

[vBenthem97] J. van Benthem, Dynamic Bits and Pieces, LP-97-01, Research Report of the Institute for Logic, Language and Information, 1997.

[BGG96] E. Börger, E. Grädel, Y. Gurevich, The Classical Decision Problem, Springer Verlag, Berlin Heidelberg, 1996.

[CL73] C-L. Chang, R. C-T. Lee, Symbolic Logic and Mechanical Theorem Proving, Academic Press, New York, 1973.

[DG79] B. Dreben, W.D. Goldfarb, The Decision Problem, Solvable Classes of Quantificational Formulas, Addision-Wesley Publishing Company, Inc. 1979.

[FLTZ93] C. Fermüller, A. Leitsch, T. Tammet, N. Zamov, Resolution Methods for the Decision Problem, Lecture Notes in Artificial Intelligence 679, Springer Verlag, 1993.

[GMW97] H. Ganzinger, C. Meyer, C. Weidenbach, Soft Typing for Ordered Resolution, in CADE 14, pp. 321-335, 1997.

[Graedel97] E. Grädel, On the Restraining Power of Guards, manuscript, 1997.

[Joyner76] W. H. Joyner, Resolution Strategies as Decision Procedures, J. ACM 23, 1 (July 1976),l pp. 398-417, 1976.

[McCune95] W. W. McCune, Otter 3.0 Reference Manual and Guide, Argonne National Laboratory, Mathematics and Computer Science Division, can be obtained from **ftp.mcs.anl.gov**, directory **pub/Otter**, 1995.

[Niv93] Generic Resolution in Propositional Modal Systems, in LPAR 93, Springer Verlag Berlin, 1993.

[Niv95] H. de Nivelle, Ordering Refinements of Resolution, Ph. D. Thesis, Delft University of Technology, 1995.

[OWBach95] H-J. Ohlbach, C. Weidenbach, A note on Assumptions About Skolem Functions, Journal of Automated Reasoning 15, Vol. 2, pp. 267-275, 1995.

[Robinson65] J. A. Robinson, A Machine Oriented Logic Based on the Resolution Principle, Journal of the ACM, Vol. 12. pp. 23-41, 1965

[ATP97] The CADE-13 Automated Theorem Proving System Competition, Journal of Automated Reasoning Special Issue, Vol. 18, No. 2, Edited by G. Sutcliffe and C. Suttner, 1996.

[Tammet90] T. Tammet, The Resolution Program, Able to Decide some Solvable Classes, in COLOG-88, Springer LNCS, pp. 300-312, 1990.

[Weid96] C. Weidenbach, (Max-Planck-Institut für Informatik), The Spass & Flotter Users Guide, Version 0.55, can be obtained from **ftp.mpi-sb.mpg.de**, directory **pub/SPASS**, 1997.

[Zamov72] N.K. Zamov, On a Bound for the Complexity of Terms in the Resolution Method, Trudy Mat. Inst. Steklov 128, pp. 5-13, 1972.